Enhanced Content Specification

Join the ATVEF!

Status of This Document

The Advanced Television Enhancement Forum (ATVEF) is a cross-industry group formed to specify a single public standard for delivering interactive television experiences that can be authored once using a variety of tools and deployed to a variety of television, set-top, and PC-based receivers. This document specifies the content formats and delivery mechanisms that provide the kind of enhanced television experience that will meet the needs of the industry.

The Enhanced Content Specification is a foundation specification, defining fundamentals necessary to enable creation of HTML-enhanced television content so that it can be reliably broadcast across any network to any compliant receiver. The scope is narrowly defined as we strive to build agreement across the industries that are key to the success of enhanced television.

Comments may be sent to info@atvef.com. For additional information, please visit the ATVEF Web site at http://www.atvef.com/.

Abstract

The ATVEF specification for enhanced television programming uses existing Internet technologies. It delivers enhanced TV programming over both analog and digital video systems using terrestrial, cable, satellite and Internet networks. The specification can be used in both one-way broadcast and two way video systems, and is designed to be compatible with all international standards for both analog and digital video systems.

The ATVEF specification consists of three parts:

1. Content specifications to establish minimum requirements for receivers.
2. Delivery specifications for transport of enhanced TV content.
3. A set of specific bindings.

Contents

Appendices

Overview

The ATVEF Specification was designed by a consortium of broadcast and cable networks, consumer electronics companies, television transport operators and technology companies to define a common, worldwide specification for enhanced television programming.

A central design point was to use existing standards wherever possible and to minimize the creation of new specifications. The content creators in the group determined that existing web standards, with only minimal extensions for television integration, provide a rich set of capabilities for building enhanced TV content in today's marketplace. The ATVEF specification references full existing specifications for HTML, ECMAScript, DOM, CSS and media types as the basis of the content specification. Section one of this document lists the minimal requirements for content support for compliant receivers. The specification is not a limit on what content can be sent, but rather provides a common set of capabilities so that content developers can author content once and play on the maximum number of players.

Another key design goal was to provide a single solution that would work on a wide variety of networks. ATVEF is capable of running on both analog and digital video systems as well as networks with no video at all. The specification also supports transmission across terrestrial (over the air), cable, and satellite systems as well as over the Internet. In addition, it will also bridge between networks - for example data on an analog terrestrial broadcast must easily bridge to a digital cable system. This design goal was achieved through the definition of a transport-independent content format and the use of IP as the reference binding. Since IP bindings already exist for each of these video systems, ATVEF can take advantage of this work. Section two defines two transports - one for broadcast data and one for data pulled through a return path.

While the ATVEF specification has the capability to run on any video network, a complete specification requires a specific binding to each video network standard in order to ensure true interoperability. Section three includes two bindings—the reference binding to IP and the example NTSC binding. The IP binding is the reference binding both because it provides a complete example of ATVEF protocols and because most networks support the IP protocol. The NTSC binding is included as an example of an ATVEF binding to a specific video standard. It is not the role of the ATVEF group to define bindings for all video standards. The appropriate standards body should define the bindings for each video standard - PAL, SECAM, DVB, ATSC and others.

There are many roles in the production and delivery of television enhancements. This document refers to three key roles: content creator, transport operator, and receiver. The content creator originates the content components of the enhancement including graphics, layout, interaction and triggers. The transport operator runs a video delivery infrastructure (terrestrial, cable, satellite or other) that includes a transport for ATVEF data. The receiver is a hardware and software implementation (television, set-top box, or personal computer) that decodes and plays ATVEF content. A particular group or company may participate as one, two or all three of these roles.

1 Content Specifications

The ATVEF content specification provides content creators with a reliable definition of mandatory content support on all compliant receivers. In addition, any other kind of data content can be sent over ATVEF transport including HTML, VRML, Java, or even private data files. When a content creator wants to broadcast an enhancement to play on the maximum number of receivers, the data should conform to the content specification.

In the case where the content author knows the specific capabilities of the target receiver, data can be sent over ATVEF transport that is outside the content specification including DHTML, Java, or even private data files.

In the ATVEF specification, there is one defined content specification: level 1.0.

1.1 Content Level 1.0

1.1.1 Content Formats

The foundation for ATVEF content is existing web standards. Mandatory support is required for the following standard specifications:

- HTML 4.0 (Frameset Document Type Definition)
- CSS 1
- ECMAScript
- DOM 0

Note:
ECMAScript plus DOM 0 is equivalent to JavaScript 1.1.

Note: Receivers are required to supply 1KB for session cookies. Cookies support is not required to be persistent when a receiver is turned off.

1.1.2 Content Type Support

Because ATVEF supports one-way broadcast of data, content creators cannot customize the content for each receiver as they do today with two-way HTTP. ATVEF specifies the following base profile of supported MIME types that must be supported in each receiving implementation:

- text/html (HTML 4.0)
- text/plain
- text/css (CSS1 only)
- image/png (no progressive encoding)
- image/jpg (no progressive encoding)
- audio/basic

Support for the following widely used MIME types is currently recommended in all receiving implementations for compatibility with existing content. Support is not and content creators should take into account that the types may not be supported.

- image/gif (no progressive encoding)
- audio/wav

Please see Appendix B for additional information on content type support.

1.1.3 Integrating TV with Web Pages

Use the `"tv:"` URL to reference a broadcast television channel. The `"tv:"` URL may be used anywhere that a URL may reference an image.

Examples of `"tv:"` URL usage include the `object`, `img`, `body`, `frameset`, `a`, `div` and `table` tags. For examples with specific HTML syntax, see Appendix A.

1.1.4 The Trigger Receiver Object

TV enhancement HTML pages that expect to have triggers sent to them via an ATVEF trigger stream must use the HTML `object` tag to include a trigger receiver object on a page. The trigger receiver object, implemented by the receiver, processes triggers for the associated enhancement in the context of the page containing the object. The content type for this object is `"application/tve-trigger"`. If a page consists of multiple frames, only one may contain a receiver object.

Sample instantiation:

```
<OBJECT TYPE="application/tve-trigger" ID="triggerReceiverObj">
</OBJECT>
```

Properties:

`triggerReceiverObj.enabled`	A boolean, indicating if the triggers are enabled. The default value is true (read/write)
`triggerReceiverObj.sourceId`	A string containing the ASCII-hex encoded UUID for the announcement for this stream. `sourceID` is `null` if the `UUID` was not set for the enhancement. (read only)
`triggerReceiverObj.releasable`	A boolean indicating that the currently displayed top level page associated with the active enhancement can be released and may be automatically replaced with a new resource when a valid trigger containing a new URL is received. Such a trigger must contain a `[name:]` attribute. The default value is `false`.
`triggerReceiverObj.backChannel`	A string indicating the availability and state of a backchannel to the Internet on the current receiver. When backChannel returns "permanent" or "connected," receivers can generally perform HTTP get or post methods and expect real-time responses. When backChannel returns "disconnected," receivers can also expect to perform HTTP get or post methods but there will be an indeterminate delay while a

	connection is established. When backChannel returns "unavailable," no HTTP get or post methods can be performed. No standard behavior can be assumed when any other value is returned. Value is one of: • `permanent` --Always connected • `connected` --Currently connected, but not always • `disconnected` --Not currently connected, but can connect • `unavailable` --Never connected
`triggerReceiverObj.contentLevel`	A number that corresponds to the ATVEF content level of the receiver. For this specification, it is 1.0.

1.1.5 Triggers

Triggers are real-time events delivered for the enhanced TV program. Receiver implementations will set their own policy for allowing users to turn on or off enhanced TV content, and can use trigger arrival as a signal to notify users of enhanced content availability.

Triggers always include an URL, and may optionally also include a human-readable name, an expiration date, and a script. Receiver implementors are free to decide how to turn on enhancements and how to enable the user to choose among enhancements. Triggers that include a "name" attribute may be used to initiate an enhancement either automatically, or with user confirmation. The initial top-level page for that enhancement is indicated by the URL in that trigger. Triggers that do not include a "name" attribute are not intended to initiate an enhancement, but should only be processed as events which affect (through the "script" attribute) enhancements that are currently active. If the URL matches the current top-level page, and the expiration has not been reached, the script is executed on that page through the trigger receiver object (see Trigger

Receiver Object). When testing for a match, parameters and fragment identifiers (i.e. characters in the URL including and following the first "?" or "#" character) in an URL are ignored.

Triggers are text based, and their syntax follows the basic format of the EIA-746A standard (7-bit ASCII, the high-order bit of the first byte must be "0"). Note: The triggers follow the syntax of EIA-746A, but may be transported in multicast IP packets or other transport rather than using the EIA-608 system.

All triggers defined in this version of ATVEF are text-based and must begin with ASCII '<'. All other values for the first byte are reserved. These reserved values may be used in the future to signal additional non-text based messages. Receivers should ignore any trigger that does not begin with the '<' in the first byte.

The general format for triggers (consistent with EIA-746A) is a required URL followed by zero or more attribute/value pairs and an optional checksum:

<*url*> [*attr*$_1$: *val*$_1$][*attr*$_2$:*val*$_2$]...[*attr*$_n$:*val*$_n$][*checksum*]

- Character set: All characters are based on ISO-8859-1 character set (also known as Latin-1 and compatible with US-ASCII) in the range 0x20 and 0x7e. Any need for characters outside of this range (or excluded by attribute limits below) must be encoded using the standard Internet URL mechanism of the percent character ("%") followed by the two-digit hexadecimal value of the character in ISO-8859-1.
- The trigger begins with a required URL:

<*url*> The URL is enclosed in angle brackets (e.g. `<http://xyz.com/fun.html>`). Although any URL can be sent in this syntax, ATVEF content level 1 only requires support for http: and lid: URL schemes.

The following attribute/value pairs are defined:

[name:*string*] The **name** attribute provides a readable text description (e.g. `[name:Find Out More]`). The *string* is any string of characters between 0x20 and 0x7e except square brackets (0x5b and 0x5d) and angle brackets (0x3c and 0x3e). The name attribute can be abbreviated as the single letter "n" (e.g. `[n:Find Out More]`).

[expires:*time*] The **expires** attribute provides an expiration date, after which the link is no longer valid (e.g. `[expires:19971223]`). The *time* to the ISO-8601 standard, except that it is assumed to be UTC unless the time zone is specified. A recommended usage is the form *yyyymmdd*T*hhmmss*, where the capital letter "T" separates the date from the time. It is possible to shorten the *time* string by reducing the resolution. For example *yyyymmdd*T*hhmm* (no seconds specified) is valid, as is simply

yyyymmdd (no time specified at all). When no time is specified, expiration is at the beginning of the specified day. The expires attribute can be abbreviated as the single letter "e" (e.g. `[e:19971223]`).

[script:*string*] The **script** attribute provides a script fragment to execute within the context of the page containing the trigger receiver object (e.g. `[script:shownews()]`). The *string* is an ECMAScript fragment. The script attribute can be abbreviated as the single letter "s" (e.g. `[s:shownews()]`).An example of a script attribute used to navigate a frame within a page to a new URL: `[script:frame1.src="http://atv.com/f1"]`

- The optional checksum must come at the end of the trigger. (Note: EIA-746A requires the inclusion of a checksum to ensure data integrity over line 21 bindings. In other bindings, such as IP, this may not be necessary, and is not required.)

[*checksum*] The checksum is provided to detect data corruption. To compute the checksum, adjacent characters in the string (starting with the left angle bracket) are paired to form 16-bit integers; if there are an odd number of characters, the final character is paired with a byte of zeros. The checksum is computed so that the one's complement of all of these 16-bit integers plus the checksum equals the 16-bit integer with all 1 bits (0 in one's complement arithmetic). This checksum is identical to that used in the Internet Protocol (described in RFC 791); further details on the computation of this checksum are given in IETF RFC 1071. This 16-bit checksum is transmitted as four hexadecimal digits in square brackets following the right square bracket of the final attribute/value pair (or following the right angle bracket if there are no attribute/value pairs). The checksum is sent in network byte order, with the most significant byte sent first. Because the checksum characters themselves (including the surrounding square brackets) are not included in the calculation of the checksum, they must be stripped from the string by the receiver before the checksum is recalculated there. Characters outside the range 0x20 to 0x7e (including the second byte of two-byte control codes) shall not be included in the checksum calculation.

Other attributes could be defined at a later date. However, all other single character attribute names are reserved. Receivers should ignore attributes they do not understand.

Using the description above, all the following are valid trigger strings:

<http://xyz.com/fun.html>

`<http://xyz.com/fun.html>[name:Find out More!]`

`<lid://xyz.com/fun.html>[n:Find out More!]`

`<lid://xyz.com/fun.html>[n:Fun!][e:19991231T115959]`
`[s:frame1.src="http://atv.com/frame1"]`

`<http://www.newmfr.com>[name:New][C015]`

Note: If a trigger does not contain a `[name:]` attribute, the enhancement referenced by the trigger should not be presented to the user.

1.1.6 The Local Identifier URL Scheme ("lid:")

Content delivered by a one-way broadcast is not necessarily available on-demand, as it is when delivered by HTTP or FTP. For such content, it is necessary to have a local name for each resource. To support cross-references within the content (for use in hyperlinks or to embed one piece of content in another), these local names must be location-independent.

The `"lid:"` URL scheme enables content creators to assign unique identifiers to each resource relative to a given namespace. Thus the author can establish a new namespace for a set of content and then use simple, human-readable names for all resources within that space. The `"lid:"` scheme is used by the `"Content-Location:"` field in the UHTTP resource transfer header to identify resources that should be stored locally by a broadcast capable receiver platform and are not accessible via the Internet.

The syntax of the `"lid:"` URL is as follows:

lid://{namespace-id}[/{resource-path}]

The `{namespace-id}` specifies a unique identifier (e.g. UUID or a domain name) to use as the namespace for this content or as a root for the URL. The `{resource-path}` names a specific resource within the namespace, and must follow the generic relative URL syntax. As with all URL schemes that support the generic relative URL syntax, this path component can be used alone as a relative URL, where the namespace is implied by a base URL specified for the content through other means.

While all compliant implementations of enhanced TV receivers support absolute URLs within the UHTTP header and broadcast triggers, some implementations may not correctly process absolute URLs using the `"lid:"` scheme within HTML content. To ensure that HTML content is correctly interpreted by these receiving platforms, content should use only relative URLs in their HTML. Triggers use the full `"lid:"` URL to load the top level HTML page and that page uses relative URLs to refer to other resources.

Some examples:

- `lid://unique2345@blahblah.com/rootimage.jpg`

- `lid://xyz.com/myshow/episode100/george.html`
- `lid://12abc554c3d3dd3f12abc554c3d3dd3f/logos/ourlogo.gif`

The first example uses a RFC 822 message-id style unique id, the second one uses a domain name as a unique identifier, and the third uses a text encoding of an UUID. Each is a valid mechanism for describing a "`lid:`" namespace.

1.1.7 Content Caching

Receivers must be able to support one megabyte (1 MB) of cached simultaneous content. Content creators who want to reach the maximum number of receivers should manage their content to require a high-water mark of simultaneous cached content of 1 MB or less. The specific cache size required for each enhancement must be specified in the announcement.

`tve-size` represents the maximum size cache needed to hold content for the current page at any time during the program and also all pages reachable by local links. It is the high water mark during the program, not the total content delivered during the program. Size is measured as the size when the content is delivered (after decompression for content sent using gzip or other compression techniques).

1.2 Additional Content Levels

In the ATVEF spec, there is only one defined content specification--level 1.0. The content level of the client is available via ECMAScript using the receiverObj.contentLevel property, and can be used in announcements. Possible directions for future content levels include Dynamic HTML, synchronized multimedia, 3-D rendering, tuning, XML, Java, and higher-quality audio among others.

---

2 Transport Specifications

The display of enhanced TV content consists of two steps: delivery of data resources (e.g. HTML pages) and display of named resources synchronized by triggers. All forms of ATVEF transport involve data delivery and triggers. The capability of networks for one-way and/or two-way communication drives the definition of two models of transport.

ATVEF defines two kinds of transport. Transport A is for delivery of triggers by the forward path and the pulling of data by a (required) return path. Transport B is for delivery of triggers and data by the forward path where the return path is optional.

2.1 Transport Type A: Return-path Data

Most broadcast media define a way for data service text to be delivered with the video signal. In some systems, this is called closed captioning or text mode service; in other systems, this is called teletext or subtitling. For the sake of this discussion, triggers delivered over such mechanisms will be generically referred to as **broadcast data**

triggers.

Some existing broadcast data services provide a mechanism for trigger delivery, but not resource deliver, due to limited bandwidth. Content creators may encode broadcast data triggers using these. Broadcast data streams only contain broadcast data triggers so there is no announcement or broadcast content delivery mechanism. Because there are no announcements, the broadcast data service stream is considered to be implicitly announced as a permanent session.

In addition to the other attributes used in triggers (see section 1.1.5), ATVEF transport type A triggers must contain an additional attribute, "`tve:`". The "`tve:`" attribute indicates to the receiver that the content described in the trigger is conformant to the ATVEF content specification level. For example, `[tve:1.0]`. The "`tve:`" attribute can be abbreviated as the single letter "v". The version number can be abbreviated to a single digit when the version ends in ".0" (e.g. `[v:1]` is the same as `[tve:1.0]`). The "`tve:`" attribute is equivalent to the use of "`type:tve`" and "`tve-level:`" in SAP/SDP announcements in the transport type B IP multicast binding. This attribute is ignored if present in a trigger in transport B since these values are set in transport type B in the announcement. If the "`tve:`" attribute is not present in a transport type A trigger, the content described in the trigger is not considered to be ATVEF content.

Television transport operators should use the standard mechanisms for broadcast data trigger transmission for the appropriate medium (EIA, ATSC, DVB, etc.). It is assumed that when the user tunes to a TV channel, the receiver locates and delivers broadcast data triggers associated with the TV broadcast. Tuning and decoding broadcast data triggers is implementation and delivery standard specific and is specified in the appropriate ATVEF binding. A mechanism must be defined for encoding broadcast data triggers for each delivery standard. For example in the NTSC binding, the broadcast data trigger syntax is encoded on the Text2 (T2) channel of line 21 using the EIA-746A system.

Because there is no content delivery system, broadcast data triggers usually require two-way Internet connections to fetch content over HTTP.

Note: Television transport operators and content creators need to plan to handle the scalability issues associated with large numbers of HTTP requests responding at roughly the same time to broadcast triggers.

2.2 Transport Type B: Broadcast Data

Transport type B is for true broadcast of both the resource data and triggers. As such, transport type B can run on TV broadcast networks without Internet connections, unlike transport type A. An additional Internet connection allowing a return path can be added to provide two way capabilities like e-commerce or general Web browsing.

Transport type B uses announcements to offer one or more enhancements of a TV channel. An announcement specifies the location of both the resource stream (the files that provide content) and the trigger stream for an enhancement. Multiple enhancements can be offered as choices that differ on characteristics like language or required cache size or bandwidth. In addition to locating the files and trigger streams, announcements must be able to provide the following information: language, start and stop times, bandwidth, peak storage size needed for incoming resources, ATVEF content level the resources represent, an optional UUID that identifies the content, an optional string that identifies the broadcast channel for systems that send ATVEF content

separately from the audio/video TV broadcast. The receiver must be able to start receiving data from only the description broadcast in the announcement.

Transport type B also requires a protocol that provides for delivery of resources. In one way broadcast systems, this is a one way resource transfer protocol that allows for broadcast delivery of resources. The resource delivered, no matter what the resource transfer method, must include HTTP headers to package the file as described in Appendix C on the resource transfer protocol. All resources delivered using resource transfer are named using URLs. These resources are then stored locally, and retrieved from this local storage when referenced using this same URL. All receivers must support local storage and retrieval of content using the `"lid:"` URL scheme (see section 1.1.6) and the familiar `"http:"` URL scheme. When `"lid:"` is used, the resources are delivered only through broadcast and are not available on demand. When "http:" is used, the resources that are delivered through broadcast also exist on the World Wide Web and can be requested from the appropriate server using standard HTTP. Sending `"http:"` resources using resource transfer effectively pre-loads the local cache, thus avoiding large numbers of simultaneous hits on Web servers when those same resources are requested by many receivers. Furthermore, this mechanism allows receivers to view the same content that appears on the Web even when no Internet connection is available. Content creators can freely mix resources that use either the `"lid:"` or `"http:"` schemes in the same enhanced broadcast. Because the underlying resource transfer protocol is not limited to carrying resources named by any particular URL scheme, some receivers will store and retrieve content named using other URL schemes, such as `"ftp:"`, as well as the required `"lid:"` and `"http:"`.

Transport type B uses the same syntax for triggers as type A, described in section 1.1.5.

The "ATVEF Reference Binding for IP Multicast" describes three protocols based on IP multicast transmission for each of the three data streams: 1) announcements; 2) triggers; and 3) one-way resource transfer.

2.3 Simultaneous Support of Transports A and B

A single video program may contain both transport type B (e.g. IP) and transport type A (e.g. broadcast data triggers) simultaneously. This is advantageous in order to target both IP-based receivers as well as receivers that can only receive broadcast data triggers.

Receivers may choose to support only IP based trigger streams and ignore broadcast data triggers, or receivers may support broadcast data triggers in the absence of IP based triggers, or receivers may support broadcast data triggers and IP based triggers simultaneously. For receivers that provide simultaneous support, ATVEF specifies the following behavior, which is identical to the treatment of IP based triggers on an active stream.

When a broadcast data trigger is encountered, its URL is compared to the URL of the current page. If the URLs match and the trigger contains a script, the script should be executed. If the URLs match but there is no script, the trigger is considered a re-transmission of the current page and should be ignored. If the URLs do not match and the trigger contains a name, the trigger is considered a new enhancement and may be offered to the viewer. If the URLs do not match and there is no name, the trigger should be ignored.

3 ATVEF Bindings

An ATVEF binding is a definition of how ATVEF runs on a given network. The binding may support either or both Transport types A and B. Having one standard ATVEF binding for each network is necessary so that receivers and broadcast tools can be developed independently.

The measure of a sufficient ATVEF binding is that all the data needed to build a compliant, interoperable receiver for a given network should be contained in the ATVEF spec, the network spec and the ATVEF network binding, if needed. Put another way, the ATVEF binding provides the glue between the network spec and the ATVEF spec, in cases where the network specification doesn't contain all the necessary information.

ATVEF defines the Binding to IP as the *reference binding*. This is because IP is available to run over virtually any kind of network in existence. That means that one approach to building an ATVEF binding for a particular network is to simply define how IP is run on that network associated with a particular video program. The IP Binding can also be used as a model for a complete, compliant and efficient ATVEF binding.

This section also includes an example of a binding to a specific network standard--the ATVEF Binding to NTSC. This binding can be used as a model for how to build an ATVEF binding to a specific video standard. The example NTSC binding defines transport type A using an NTSC-specific method and defines transport type B using the IP reference binding. It is not the role of the ATVEF group to define bindings for all video standards. The appropriate standards body should define the bindings for each video standard--PAL, SECAM, DVB, ATSC and others.

3.1 ATVEF Binding to IP Multicast (Reference Binding)

IP multicast is the mechanism for broadcast data delivery. Content creators should assume IP addresses may be changed downstream, and therefore should not use them in their content. The transport operator is only responsible for making sure that an IP address is valid on the physical network where they broadcast it (not for any re-broadcasting). When possible, content creators should use valid IP multicast addresses to minimize the chance of collisions. Some systems may have two-way Internet connections. Capabilities in those systems are outside the scope of this document and are described by the appropriate Internet standards.

Transport operators should use the standard IP transmission system for the appropriate medium (IETF, ATSC, DVB, etc.). It is assumed that when the user tunes to a TV channel, the receiver automatically locates and delivers IP datagrams associated with the TV broadcast. The mechanism for tuning video and connecting to the appropriate data stream is implementation and delivery standard specific and is not specified in this framework.

3.1.1 Announcement Protocol

Announcements are used to announce currently available programming to the receiver. The IP multicast addresses and ports for resource transfer and for triggers are announced using SDP announcements (RFC 2327). The SDP Header is preceded by an 8-byte SAP header. SAP is still in Internet Draft form, but the non-optional first 8 bytes are

stable (http://www.ietf.org/html.charters/mmusic-charter.html). Announcements are sent on a well-known address (224.0.1.113) and port (2670). This address and port have been registered with the IANA.

`v=0`	SDP Version, required to be 0.
`o=username` *sid* *version* `IN IP4` *ipaddress*	Owner & session identifier, defined as usual in SDP spec. Username is "-", network type is IN, address type is IP4. SessionID identifies an announcement for a particular broadcast (it can be a permanent announcement for all programming on a broadcast channel or for a particular show). Version indicates the version of the message. These values allow receivers to match a message to a previous message and know whether it has changed. Session ID and Version should be NTP values as recommended in SDP.
s=name	Session name, required as in SDP spec.
`i=, u=`	Optional, as in SDP spec.
`e=, p=`	E-mail address or phone number, at least one required in SDP spec.
`b=CT:`*number*	Optional in SDP spec, but Required here. Bandwidth in kbps as in the SDP spec. Bandwidth of the broadcast data can be used by receivers to choose among multiple versions of enhancement data according to the bandwidth the receiver can handle.
`t=`*start stop*	As in SDP spec gives start and stop time in NTP format. With programs stored on tape, at times it will not be possible to insert new announcements, so start times on tape could be incorrect. In this case, the start time should be set to the original broadcast time and the stop time set to 0. This is the standard for an unbounded session. Assumptions are then made about the stop time (see RFC 2327). A new announcement for a new program for the same broadcast station replaces the previous one. It is preferred that a tool read the tape and generate announcements with correct start and stop times, but not required. Content creators can choose to use only a station ID and not provide information about individual programs.
`a=UUID:`*UUID*	Optional. The UUID should uniquely identify the enhancement (for example, a different UUID for each program), and can be accessed using the trigger receiver object. In analog TV and many types of digital TV broadcast data is tied tightly to A/V. Each virtual channel has its own private network associated with it. In other systems,

	enhancements for many virtual channels can be carried on the same network. These systems can use the UUID to link a TV broadcast with a particular enhancement. How that association is indicated is beyond the scope of this document. One technique would be to place the UUID in electronic program guide information. Use ASCII HEX to encode UUIDs.
`a=type:tve`	Required. Indicates to the receiver that the announcement refers to an ATVEF
a=*lang*, a=*sdplang*	Optional, as in SDP spec.
`a=tve-type:<types>`	Optional. `tve-type:` specifies an extensible list of types that describe the nature of the enhancement. It is a session-level attribute and is not dependent on charset. `a=tve-type:primary` Optional. `tve-type:primary` specifies that this will be the primary enhancement stream associated with the currently playing video program whenever this enhancement's trigger stream is active. If `tve-type:primary` is not specified, the TVE stream is never the primary enhancement stream associated with video. This, like all `tve-type:` attributes, is a session level attribute. This attribute can be used by receivers to implement automatic loading of primary video enhancement streams. The actual display of and switching between enhancement streams is handled by the trigger streams.
a=tve-size:*Kbytes*	Required. `tve-size:` provides an estimate of the high-water mark of cache storage in kilobytes that will be required during the playing of the enhancement. This is necessary so that receivers can adequately judge whether or not they can successfully play an enhancement from beginning to end.
a=tve-level:*x*	Content level identifier, where x is 1.0 for this version of the framework (optional, default is 1.0).
a=tve-ends:*seconds*	Optional, specifies an end time relative to the reception time of the SDP announcement. *Seconds* is the number of seconds in the future that this announcement is valid. *Seconds* may change (count down) as an announced session progresses. This attribute, when present, overrides the default assumptions for end times in unbounded announcements.
m=data *portvalue/2*	As in SDP spec. Compact form specifying 2 ports on same address

```
tve-file/tve-
trigger
c=IN IP4
ipaddress/ttl
```

When there are multiple alternative enhancement streams for the same video program, they must all be announced at the media level of the same SDP announcement. All enhancement streams announced in the same SDP announcement are considered to be mutually exclusive variants of the primary enhancement stream. The receiver can choose between them based on media level attributes. For example, the `a=lang` field can be used at the media level to choose between language variants of the primary enhancement.

Each media section for the tve-file media type begins the next enhancement definition.

A longer form is available if the content creator or transport operator wants to use different IP addresses and ports for the data stream and trigger stream:

`m=data` *portvalue* `tve-file` `c=IN IP4` *ipaddress/ttl*	Alternative form for specifying addresses and ports (for file protocol, as in SDP spec)
`m=data` *portvalue* `tve-trigger` `c=IN IP4` *ipaddress/ttl*	For control protocol, as in SDP spec.

Announcement Example:

```
v=0
o=-2890844526 2890842807 IN IP4 tve.niceBroadcaster.com
s=Day & Night & Day Again
i=A very long TV Soap Opera
e=help@niceBroadcaster.com
a=UUID:f81d4fae-7dec-11d0-a765-00a0c91e6bf6
a=type:tve
a=tve-level:1.0
t=2873397496 0
a=tve-ends:30000
a=tve-type:primary
m=data 52127/2 tve-file/tve-trigger
c=IN IP4 224.0.1.112/127
b=CT:100
a=tve-size:1024
m=data 52127/2 tve-file/tve-trigger
c=IN IP4 224.0.0.1/127
b=CT:1024
a=tve-size:4096
```

3.1.2 Trigger Protocol

The trigger protocol carries a single trigger in a single UDP/IP multicast packet. Triggers

are real-time events broadcast inside IP multicast packets delivered on the address and port defined in the SDP announcement for the enhanced TV program (see Announcements). The trigger protocol is thus very lightweight in order to provide quick synchronization.

3.1.3 Resource Transfer: UHTTP

A one-way IP multicast based resource transfer protocol, the Unidirectional Hypertext Transfer Protocol (UHTTP) is defined in Appendix C. UHTTP is a simple, robust, one-way resource transfer protocol that is designed to efficiently deliver resource data in a one-way broadcast-only environment. This resource transfer protocol is appropriate for IP multicast over television vertical blanking interval (IPVBI), in IP multicast carried in MPEG-2, or in other unidirectional transport systems.

Web pages and their related resources (such as images and scripts) are broadcast over UDP/IP multicast along with their related TV signal. An announcement broadcast by the TV station tells the receiver which IP multicast address and port to listen to for the data. The only data broadcast to this address and port are resources intended for display as Web content.

While HTTP headers preceding resource content are optional in the UHTTP protocol, they are required when the protocol is used for ATVEF enhanced TV. Compliant receivers must support content encodings of "gzip" as specified by the "Content-Encoding" HTTP header field.

3.2 ATVEF Binding to NTSC

In NTSC, ATVEF data is broadcast by encoding bytes in the vertical blanking interval of individual video fields. Two different techniques are used for broadcasting data using ATVEF transport A and ATVEF transport B.

3.2.1 Transport A: VBI Line 21

ATVEF triggers are transmitted on VBI Line 21 of the NTSC signal using the T-2 service as specified in EIA-608. This encoding is consistent with the EIA-746A specification which describes how to send URLs and related information on VBI line 21 of an NTSC channel, without interfering with other data (e.g., closed captions) also sent on that line. The checksum described in the ATVEF trigger definition is required in the Transport A ATVEF Binding to NTSC.

Note that, as specified in the ATVEF trigger definition, triggers are encoded using ISO-8859-1 and not the EIA-608 character set. (While most characters are the same in both encodings, a few codes have different meanings.)

ATVEF trigger length should be kept as short as possible. ATVEF trigger transmissions should be limited to 25% of the total field 1 bandwidth, even if more bandwidth is available after captioning, to allow for other downstream services.

3.2.2 Transport B: IP over VBI

IP datagrams should be sent according to the specification drafted by the IP over VBI working group of the Internet Engineering Task Force (see http://www.ietf.org/html.charters/ipvbi-charter.html. Note that this specification is currently in late draft stage, but is expected to be completed and published as a standards-track document in the coming weeks. In NTSC, the NABTS (rather than WST) byte encoding should be used.

ATVEF IP streams should be sent on the packet addresses 0x4b0 through 0x4bf. Other packet addresses may be used, but receivers are only required to handle IP datagrams arriving using packet addresses 0x4b0 through 0x4bf.

Appendix A: Examples of Integrating TV with Web Pages

The following examples describe how to achieve common design goals for integrating TV and Web pages. This list is meant to be illustrative rather than exhaustive. The "tv:" URL may be used anywhere that an image URL is also appropriate.

Examples are presented in both HTML 3.2 and HTML 4.0 since the HTML 4.0 specification recommends that tools supporting HTML 4.0 continue to support HTML 3.2.

1. How to place TV in a web page (using <OBJECT> and <IMG> tags)

The OBJECT and IMG tags are used to place the TV picture in a web page, for example:

```
<object data="tv:" width="60%" height="60%">
<img src="tv:" width=320 height=240>
```

2. How to place TV in a web page that uses tables (using <TABLE> tags)

The TD tag can be used to place the TV picture as the background of a table cell, for example:

```
<td width=320 height=240 style="background: url(tv:)">
    Here is content that is overlaid on top of the
    TV picture inside this table cell.
</td>
```

3. How to overlay a web page over a TV background (using <BODY> tag)

The BODY tag is used to specify TV as a full screen background of the web page, for example:

HTML 3.2 syntax: `<body background="tv:">`
HTML 4.0 syntax: `<body style="background: url(tv:)">`

4. How to overlay a frame-based web page over a TV background (using <FRAMESET> tag)

Many ATVEF web pages will be frame-based rather than body tag based. This will allow

the program to change the displayed web page while maintaining the same URL for a series of triggers. Since an HTML document that contains a FRAMESET tag cannot contain a BODY tag, it is necessary to specify "tv:" on a FRAMESET when full screen TV is desired beneath the frames, for example:

```
<frameset style="background: url(tv:)" cols="200,*">
```

Each frame in the frameset that wants the full screen TV to show through must specify a transparent background color in the BODY tag of the frame's HTML document, for example:

HTML 3.2 syntax: `<body bgcolor="transparent">`
HTML 4.0 syntax: `<body style="background: transparent">`

5. How to transition from a web page back to full-screen TV (using <A> tag)

Finally, the use of "tv:" as the href of an anchor tag allows for hyperlinking to full screen TV, for example:

```
<a href="tv:">Click here to return to TV</a>
```

Appendix B: Content Format Notes

Content creators should use the content formats specified in section 2.1. This will guarantee that the content will play on the largest number of ATVEF receivers since support for this set of content types is mandated.

Image content should be sent using PNG image format whenever possible. Currently, PNG does not support animation or high ratio (lossy) compression for natural images. When these features are available in PNG or another open standard, they will most likely be rolled into an ATVEF content level. In the meantime, many current web browsers support these features through GIF and JPEG. Content creators may wish to employ GIF for animated images and JPEG for high-compression images with some confidence that those image types will be supported on many platforms.

With any image format, it is recommended that progressive rendering features be avoided (e.g. progressive PNG, progressive JPEG, interlaced GIF). Progressive rendering allows a client to display a low quality version of the image at first, improving quality as the image is downloaded. Progressive rendering may not be supported on some small footprint receivers.

Audio content should be sent with the standard audio/basic format to reach the widest number of ATVEF receivers. The audio/basic format is a simple audio format of single channel audio encoded using 8 bit ISDN mu-law [PCM] at a sample rate of 8000 Hz. For higher-quality audio needs, content creators may wish to use other widely supported forms of the WAV and AIFF formats with some confidence that those audio types will be supported on many platforms.

Appendix C: The Unidirectional Hypertext Transfer Protocol (UHTTP)

Overview

The Unidirectional Hypertext Transfer Protocol, or UHTTP, is a simple, robust, one-way data transfer protocol that is designed to efficiently deliver resource data in a one-way broadcast-only environment. This transfer protocol is appropriate for one-way IP multicast over television vertical blanking interval (IP/VBI) or other unidirectional transport systems.

This section describes the format of the message packets that carry UHTTP data. It describes the information needed to create the messages using the protocol on the broadcast side and to turn those messages back into resources on the receiving side.

Resources sent using the UHTTP protocol are divided into a set of packets, encapsulated in UDP. Typically, these packets may be delivered via multicast IP, but this is not required. Each packet contains enough header information to begin capturing the data at any time during the broadcast, even midway through the transfer. This header contains an identifier (in the form of an UUID) that uniquely identifies the transfer, and additional information that enables the receiver to place the data following the header in the appropriate location within the transfer. Additional information indicates to the receiver how long to continue listening for additional data.

UHTTP includes the ability to gather segments over multiple retransmissions to correct for missing packets. It is also possible to group resources together for all-or-none delivery within a UHTTP transfer. The protocol also includes a forward error correcting mechanism which provides for the ability to restore missing data in the event of limited packet loss.

Data Transfer Features Enabled by the UHTTP Protocol

Robust Delivery: Gathering data over multiple transmissions

Data can be resent via UHTTP using the same globally unique `TransferID`. The data is delivered as individual segments, each of which is in a UDP message, potentially delivered via IP multicast. Information in the header allows a receiving application to receive segments out of order or multiple times. If the transfer data is sent repeatedly, the receiving service can fill in missing ranges using these retransmissions. This provides robust (though not necessarily reliable) data delivery. Additionally, forward-error correction (FEC), using an XOR algorithm, provides for recovery of some missing segments in the face of segment loss without re-transmission.

Meta-information in the form of HTTP-style headers

The protocol provides for the inclusion of HTTP-style headers preceding the resource data. These headers may include information describing the content type of the resource and content location in the form of a URL. It may also be used to describe groups of resources as a multipart construction. Other meta-information, including date stamping and expiration dates, may be used to provide additional information about the resource content.

UHTTP Header Details

The UHTTP header is at the start of every UHTTP IP/UDP multicast payload. All values are network byte order. The fields are as follows:

Name	Size	Description
Version	5 bits	Describes the version of the protocol. The protocol described here is version 0.
ExtensionHeader	1 bit	When set, this bit indicates that one or more extension header fields are present.
HTTPHeadersPrecede	1 bit	A bit flag that, when set to 1, indicates that HTTP-style headers precede the resource data. These HTTP-style headers are considered part of the data when calculating the ResourceSize and SegStartByte fields, as well as for forward error correction. This bit must be set in all packets associated with a UHTTP transfer when HTTP-style headers precede the data. When set to zero, no HTTP-style headers precede the resource data.
CRCFollows	1 bit	When the CRCFollows bit is set to 1, a 32 bit CRC is calculated and can be used to detect possible corruption in the data delivered via UHTTP. Using the MPEG-2 CRC algorithm, the CRC is calculated on the complete data, including HTTP-style headers, if any. It is then appended to the end of the data in the last logical packet. This CRC field is considered part of the data for the purposes of calculating the resource length and calculating the forward error correction. The bit must be set in all packets associated with a UHTTP transfer when a CRC is used.
PacketsInXORBlock	1 byte	Describes the number of packets in a forward error correction block, including the forward error correction packet. Set to zero when no forward error correction is used.
	2	Time in seconds over which the

RetransmitExpiration	bytes	resource may be retransmitted. This indicates how long the receiving software should wait to try to recover missing packets that follow in retransmissions of the same resource. This allows a resource to be carouseled, or sent repeatedly to increase the chances of delivery without missing segments. Set to zero if the resource will not be retransmitted. Set to maximum if the software should continue listening. The RetransmissionExpiration field should be updated to remain accurate during retransmissions, including the current transmission.
TransferID	16 bytes	Globally unique identifier (UUID) for the UHTTP transfer. This ID allows receiving software to identify which segments correspond to a given transfer, and determine when re-transmission occurs.
ResourceSize	4 bytes	Size of the complete resource data itself (excluding segment headers, XOR segments and padding for exclusive-or correction). This length does include the length of the HTTP-style headers, if any, as well as the 4-byte CRC, if the CRCFollows bit is set to 1.
SegStartByte	4 bytes	Start byte in the transfer for this data segment. When XOR data is used to replace missing packets, SegStartByte includes the XOR data as well as the resource data, and optional HTTP-style headers and CRC. This allows for determining where all packets fit regardless of delivery order. The exclusive-or correction packet looks like any other UHTTP packet. Its data payload is simply the exclusive-or of a number of packets that precede it in order in the data. The number of packets in an XOR block is specified in the PacketsInXORBlock field described above.

Extension Headers		Extension headers, if any.
Data Payload		The data payload for the UHTTP transfer, including HTTP-style headers, if any, and body.
Total Length:	28 bytes	

The UDP packet data length for the enclosing UDP packet is used to determine the length of the segment. It is permissible to send a packet that contains UHTTP header (and optional extension headers), but without any data. If no data is included, then the SegStartByte field is ignored.

UHTTP Extension Headers

If the ExtensionHeader flag is set in a UHTTP packet, additional optional header fields are present. These fields appear directly after main UHTTP header. Extension headers are optional on a packet-by-packet basis, and may appear on none, some or all of the UHTTP packets transmitted, depending on the ExtensionHeaderType. This specification defines a single extension header type, HTTPHeaderMap (defined below). Any extension headers with an unknown type should be ignored by receivers. The format for the fields within a UHTTP extension header are as follows:

Name	Size	Description
ExtensionHeaderFollows	1 bit	When 1, this field indicates that another extension header follows this one. When 0, the UHTTP data payload follows this extension header.
ExtensionHeaderType	15 bits	Identifies the extension header type. (1 = HTTPHeaderMap, all other values reserved).
ExtensionHeaderDataSize	2 bytes	Describes the length of the complete Extension Header data in bytes. Zero indicates that there is no ExtensionHeaderData following.
ExtensionHeaderData		The variable length data for this extension header. The length of the ExtensionHeaderData field is indicated by the ExtensionHeaderDataSize.

If the ExtensionHeaderFollows bit is set, then another ExtensionHeader follows this header. If the bit is cleared, then the UHTTP data payload follows the ExtensionHeaderData (if any) immediately.

HTTPHeaderMap Extension Header

One `ExtensionHeaderType` is defined for this specification. When `ExtensionHeaderType` is set to a value of 1, then the `ExtensionHeaderData` field contains an `HTTPHeaderMap`. A `HTTPHeaderMap` extension header may optionally be included whenever the UHTTP transfer contains HTTP-style header information (as indicated by the `HTTPHeadersPrecede` bit in the main UHTTP header). If `HTTPHeaderMap` extension headers are used, they should be included in every packet in a UHTTP transfer that contains header, body or forward-error correction (FEC) data.

The `HTTPHeaderMap` consists one or more sets of the following fields:

Name	Size	Description
`HTTPHeaderStart`	4 bytes	This field indicates an offset into the UHTTP data, in bytes, where a HTTP-style header is found. The offset is calculated from the beginning of the corrected UHTTP data, and does not include the FEC data when the FEC mechanism is used.
`HTTPHeaderSize`	4 bytes	This field indicates the length of the HTTP-style header, in bytes, including the HTTP-style header fields, the terminating pair of newline characters, and any preceding multipart boundary lines.
`HTTPBodySize`	4 bytes	This field indicates the length of the data body, in bytes, associated with the HTTP header described in this map entry.

When the UHTTP transfer consists of a single (i.e. non-multipart) resource, a single 12 bytes set of `HTTPHeaderMap` fields is present in the `HTTPHeaderMap`. The `HTTPHeaderStart`, in this case, will be set to zero and the `HTTPHeaderSize` will be set to the sum of the length of the HTTP-style header fields and all separating newline characters. The HTTPBodySize field will contain the size, in bytes, of the body data related to that header field.

When a UHTTP transfer contains multiple resources (as specified by a multipart content-type), multiple sets of `HTTPHeaderMap` groups may be included in the `HTTPHeaderMap` data, each indicating the offset and size of the HTTP-style headers for each resource, (including any multipart boundary lines, HTTP-style header fields and separating newline characters), as well as the size of the body relating to each header.

When including `HTTPHeaderMap` data, senders must at a minimum include `HTTPHeaderMap` entries for each HTTP-style header that is partially or completely included in a given packet. Additionally, when forward-error correction is used in UHTTP transfers that contain `HTTPHeaderMaps` extension headers, senders must include `HTTPHeaderMap` entries as extension headers in FEC-data packets for all HTTP-style header sections that may be corrected by the FEC packet. Senders are free to include additional `HTTPHeaderMap` entries in any packet beyond the minimum.

Forward Error Correction Mechanism

In addition to the ability to retransmit data and have missing segments filled in during retransmissions, this transfer protocol can also include extra data packets that can be used for simple missing packet error correction. When `PacketsInXORBlock` is set to zero, there is no exclusive-or forward error correction. When non-zero, all segments must be the same length. It is permissible to send packets with no data payload (but with UHTTP headers and optional extension headers). In this case, the packet is ignored in the calculation of forward error correction.

In blocks of `PacketsInXORBlock` equal size data segments, the last data segment in the block contains the exclusive-or of the preceding segments (`PacketsInXORBlock -1`). Each byte of the data in this "XOR segment" is the exclusive-or of the corresponding byte in each of the other segments in that data block. If the data is thought of as laid out separated into consecutive segments, then after every `PacketsInXORBlock -1` segments another segment is inserted that looks exactly like resource data and has its own position offset into the transfer like resource data. The data in that segment is the exclusive-or of the previous packets in that block. If this technique is used, the data payload of all packets must be the same size. The packet containing the end of file data (including the optional CRC) must be zero filled. Packets between this packet and the last XOR packet need not be sent since the receiver knows their contents are all zeros since it knows the overall length. If they are sent they must be zero filled after the segment header. The last XOR packet has the value `SegStartByte` calculated to be just as if zero filled extra packets were sent, but there is no requirement to send those empty packets.

To correct for a single missing packet in a block, the receiver should calculate the exclusive-or the data payload of the packets that arrived with the XOR data segment for that block. A key point is that segments can be sent in any order since each segment including the XOR segment indicate where in order they belong. By sending segments (including the XOR packets) out of order, there is protection against burst errors that lose successive packets. When re-transmitting a UHTTP transfer, a different order of segments can be used in each retransmission to avoid different types of burst errors. This protocol allows the headend (broadcast side) tools to decide how to order sending packets providing a great deal of flexibility. The receiving side does not need to know the transmission order (consistent with the fact that in general it cannot know the transmission order for IP multicast delivery). XOR data in the XOR packet is the exclusive-or of data segment contents only, including the HTTP-style header fields but not including the UHTTP header that is also in the packet.

HTTP-style headers used in UHTTP

The UHTTP transfer protocol can be used to deliver resources via a broadcast medium, which can simultaneously deliver resources, including web-related content, to large numbers of users simultaneously. HTTP-style headers are optional in UHTTP, but are required for resources intended to be interpreted as web content.

HTTP-style headers (HTTP 1.1) are required to precede the resource contents just as HTTP does when resources are sent as a response to a HTTP GET or POST command. The HTTP-style headers may provide additional information to the browser like the expiration time for the resource. The HTTP-style headers precede the body of the resource data, and are treated as part of the content. The protocol header and its version imply the equivalent HTTP response line (e.g. "HTTP/1.1 200 OK"). The header fields that are required to be supported by all receiving clients are listed below and

should be interpreted per the HTTP 1.1 specification. No assumption should be made for support of other header fields.

Supported HTTP-style headers

HTTP-style header Fields required for every resource:

```
Content-Length:
Content-Location:
```

Recommended HTTP-style header fields:

```
Content-Type:
```

Optional HTTP-style header fields:

```
Content-Base:
Content-Encoding:
Content-Language:
Content-Style-Type:
Date:
Expires:
Last-Modified:
```

Receivers will decode the headers and data and store them in a local cache system. Different platforms will have different cache sizes for storing local resources, which may or may not correspond to traditional browser caches. The use of "Content-Location:" headers with "lid:" style URLs (see The Local Identifier URL Scheme ("lid:")) is intended to mirror resource delivery to a local cache without requiring that the data be available on the web.

Receiving platforms should take into consideration that the same resources will likely be sent repeatedly to provide resources for users who tune in late. HTTP-style header fields can be examined to determine if the resource is already present, and so can be ignored. The "Date:", "Expires:", and "Last-Modified:" headers can be used to determine the lifetime of a resource in a given browser's cache.

When the "http:" scheme is specified in the URL, the HTTP-style header will contain the same information as the get response plus the "Content-Location:".

Packaging more than one resource

The HTTP "Content-Type:" field can be multipart/related. When this type is used, the HTTP-style header is ended as usual and is followed by the usual boundary structure for "multipart/related" separating multiple related resources that each use the HTTP-style header formats. This is a mechanism to package multiple related resources together in a single all-or-nothing transfer. The HTTP-style headers for individual subparts describe only the subpart, but are interpreted as per the HTTP 1.1 specification. In this case, it may be convenient to specify a "Content-Base:" for the entire package and then specify relative URLs for each of the "Content-Location:" headers for subsequent subparts.

The "multipart/related" content type should be used as per the IETF RFC 2387, with the following exceptions. The "start" and "start-info" attributes of the content-type header, which is optional in RFC 2387, are not supported.

An example using HTTP scheme URLs:

```
Content-Base: http://www.blahblah.com/
Content-Length: 3495
Content-Type: Multipart/Related; boundary=example98203804805
--example98203804805
Content-Location: http://www.blahblah.com/resource1.html
Content-Length: 495
Content-Type: text/html
Resource data for resource1.html
…<IMG src="image.jpg">…
--example98203804805
Content-Location: /image1.jpg
Content-Length: 1495
Content-Type: image/jpeg
Resource data for image1.jpg
--example98203804805
```

An identical example using "lid:" style URLs and relative URLs:

```
Content-Base: lid://unique2345@blahblah.com/
Content-Length: 3495
Content-Type: Multipart/Related; boundary=example98203804805
--example98203804805
Content-Location: resource1.html
Content-Length: 495
Content-Type: text/html
Resource data for resource1.html
…<IMG src="image.jpg">…
--example98203804805
Content-Location: image.jpg
Content-Length: 1495
Content-Type: image/jpeg
Resource data for image1.jpg
--example98203804805
```

Related Specifications

Hypertext Transfer Protocol 1.1 (IETF RFC2068): ftp://ftp.isi.edu/in-notes/RFC2068.txt

MIME multipart/related (IETF work in progress draft, replaces RFC2387): http://info.internet.isi.edu/in-notes/rfc/files/rfc2387.txt

UUIDs and GUIDs (IETF work in progress draft-leach-uuids-guids-01): *The draft is no longer available.*

MPEG-2 CRC (ISO/IEC 13818-1, Annex A: CRC Decoder Model)

Appendix D: Using Enhanced TV

Television enhancements are comprised of three related data sources: announcements (delivered via SAP), content (delivered via UHTTP), and triggers (delivered via the trigger protocol over UDP).

Announcements are broadcast on a single well-known multicast address and have a time

period for which they are valid. This time period is expressed via the "t=" and "a=tve-ends:" lines within the SDP record. Announcements also indicate the multicast address and port number that the client can listen in on to receive the content and triggers.

The announcement also contains information that the client can optionally use to help decide whether to automatically start receiving trigger and content information. This may include a=tve-type, lang=, and keywds= attributes that provide additional information to the client about the announced enhancements. For example, announcements with an optional a=tve-type:primary attribute may be used by the client to implement an "auto-play" feature. Multiple a=tve-type attributes may appear in a given announcement and are not mutually exclusive.

When the client sees a new enhancement, it knows that there will be data available on the given content and trigger addresses. The client may present the user with a choice to start receiving trigger and content information, or may do so automatically. The client implementation specifies what kind of user interface, if any, to present. After this confirmation (or automatic behavior) the client receives content and triggers, caching the content and parsing the triggers.

When the client first receives a trigger (containing a URL pointing to some enhancement content) the client may notify the user that the content is available or, alternatively, navigate to that content automatically. Clients may choose not to notify the user if they believe that they cannot display the enhancement, generally because the content referred to by the specified URL is not available.

When an enhancement has either been confirmed by the user, or has been started automatically, the enhancement is displayed. Only one enhancement may be displayed at a time. When new triggers associated with the current enhancement arrive, they are played or ignored depending on several conditions. If the URL of the trigger matches the URL of the current page and the trigger has a script attribute, the script is played; if there is no script, the trigger is ignored. If the URLs do not match and the trigger has a name attribute, the trigger is considered a new enhancement and is played, offered to the viewer, or ignored depending on other factors described below; if no name attribute, the trigger is ignored.

If a new enhancement is announced while an existing enhancement is being displayed, the client may present the user with the option to begin receiving that announcement data (content and triggers) or do so automatically. Multiple enhancements may be received simultaneously, although only one may be displayed at a time.

When the new enhancement is being received at the same time as an existing enhancement is being displayed, and the new enhancement delivers its first trigger, the client may have one of three behaviors:

- The client ignores the new enhancement trigger until the existing enhancement has been completed.
- It presents the user with the opportunity to navigate to the new enhancement.
- The client automatically navigates to the new enhancement.

It may be important for some triggers to be able to send scripts to the current enhancement without presenting the user with the opportunity to navigate to that enhancement. In this case, no [name:] attribute should be included. This allows enhancements to enforce that the user view them from the beginning and not join in

later when a subsequent trigger containing a script is received. If no `[name:]` attribute is found in the trigger, the user should not be presented with the opportunity to view the enhancement or automatically navigate there. The enhancement's data stream can be used to pre-load data by sending data before the first trigger that is sent with a `[name:]` attribute.

Content creators are encouraged to "shut down" their enhancements at the end of the related video content. This means that enhancements should navigate themselves (via trigger scripts or some other scripting mechanism) to full screen television (`"tv:"`) when the program or commercial ends. This will prevent content creators from displaying their enhancement over some unrelated broadcasts and reduce the likelihood of conflicts between producers. Content creators may wish to collaborate with the producers of subsequent programs or commercials to build a single enhancement that spans multiple video segments and may provide some enhanced user experience.

When the time period specified by the announcement is over, clients may automatically end the enhancement or allow the user to continue viewing the enhancement over potentially unrelated video.

Additionally, a property, named `.releasable` may be set on the trigger receiver object associated with the current enhancement. When set to `true`, the current enhancement associated with this trigger stream may be automatically replaced with a new enhancement if the client user interface permits this. A subsequent enhancement can become active by sending a trigger which includes a `[name:]`attribute when the current page's trigger receiver object's `.releasable` property is `true`. When `.releasable` is `false`, it is a hint from the content author that the page should not be replaced at this time. The client may decide whether or not to replace the page based on other factors as well, such as if the enhancement has run out of time and if the user has interacted with the enhancement.

Appendix E: ATVEF Example Broadcast

The following is a simple example of an ATVEF television enhancement, delivered via transport type B (multicast IP). The example consists of three parts: the announcement (announced via SDP/SAP), the content (delivered via UHTTP), and the triggers (delivered in UDP packets).

The experience consists of a screen with a 60% sized embedded live TV object, with some text below it. During the show, a trigger may arrive that will cause an image of the word "MURDER" to appear below the text. If the user chooses to click on the TV object, they will be returned to full screen video, and away from the enhanced experience.

Announcement:

The following announcement packet is sent via UDP to the multicast IP address: 224.0.1.113, port: 2670.

The announcement consists of an 8 byte SAP header followed by an SDP text payload.

The values for the SAP header fields for the announcement::

Version (3 bits)	1	SAP version
Message Type (3 bits)	0	Session description announcement packet
Encrypted (1 bit)	0	Not encrypted
Compressed (1 bit)	0	Not compressed
Authentication Length (1 byte)	0x00	No authentication
Message ID Hash (2 bytes)	0x3464	Hash of payload text
Originating Source Address (4 bytes)	209.240.195.6	IP address of originating host

Complete SAP header would be eight bytes: 0x20, 0x00, 0x34, 0x64, 0xd1, 0xf0, 0xc3, 0x06

The remaining bytes in the announcement packet would contain the following text payload:

```
v=0
o=-2890844526 2890842807 IN IP4 tve.niceBroadcaster.com
s=Day & Night & Day Again
i=A very long TV Soap Opera
e=help@niceBroadcaster.com
a=UUID:f81d4fae-7dec-11d0-a765-00a0c91e6bf6
a=type:tve
a=tve-level:1.0
a=tve-ends:1800
a=tve-type:primary
t=2873397496 0
m=data 52127/2 tve-file/tve-trigger
c=IN IP4 224.0.1.112/127
b=CT:40
a=tve-size:1024
```

These fields indicate the following:

`v=0`	SDP version zero
`o=-2890844526 2890842807 IN IP4 tve.niceBroadcaster.com`	Originating host information
`s=Day & Night & Day Again`	Session name
`i=n very long TV Soap Opera`	Session description
`e=help@niceBroadcaster.com`	Contact information about the session
`a=UUID:f81d4fae-7dec-11d0-a765-00a0c91e6bf6`	Unique identifier (UUID) for the session
`a=type:tve`	This is a television enhancement

`a=tve-level:1.0`	ATVEF content level 1.0
`a=tve-ends:1800`	Session ends 30 minutes from now
`a=tve-type:primary`	This session is the primary enhancement to the video
`t=2873397496 0`	Session began at a particular time
`m=data` *52127/2* `tve-file/tve-trigger`	File and trigger data is available on ports *52127* and *52127+1*
`c=IN IP4` *224.0.1.112*`/127`	Data will be broadcast on multicast address *224.0.1.112*
`b=CT:40`	This session will have a maximum bandwidth of 40kbps
`a=tve-size:3`	This session will require a maximum amount of caching of 3k bytes

Content:

The content data for the enhancement is delivered via UHTTP packets transmitted (as specified by the announcement) to multicast address *224.0.1.112*, to port *52127*.

This content would consist of two original source files, an HTML document and a PNG image. The experience consists of a screen with a 60% sized embedded live TV object, with some text below it. During the show, a trigger may arrive that will cause an image of the word "MURDER" to appear below the text. If the user chooses to click on the TV object, they will be returned to full screen video, and away from the enhanced experience

The first would be referred to by the URL `<lid://nicebroadcaster.com/show27/launch.html>`, and consists of the following text:

```
<HTML>
<HEAD>
<TITLE>Day & Night & Day: The Interactive Experience</TITLE>
</HEAD>

<BODY bgcolor="magenta">
<A href="tv:">
<OBJECT TYPE="application/tve-trigger" ID="triggerReceiverObj">
</OBJECT>
</A>
<BR>
<P>Welcome to the Day & Night & Day Interactive Experience</P><BR>
<P>Watch below for more information about the current scene!</P><BR>
```

```
<SCRIPT LANGUAGE="JavaScript">
function scenechange(imagename)
{
document.sceneimage.src = imagename + ".png";
}
</SCRIPT>

<IMG name="sceneimage" align="center" src="">

</BODY>
</HTML>
```

The second file consists of a PNG image, containing an image of the word "MURDER" in big red letters. It's URL will be specified as `<lid://nicebroadcaster.com/show27/murder.png>`

These files are combined together into a single multipart MIME entity, which will make up the full payload of the UHTTP transmission.

```
Content-Base: lid://nicebroadcaster.com/show27
Content-Length: 2264
Content-Type: Multipart/Related; boundary=example98203804805

--example98203804805
Content-Location: launch.html
Content-Length: 523
Content-Type: text/html

<HTML>
<HEAD>
<TITLE>Day & Night & Day: The Interactive Experience</TITLE>
</HEAD>

<BODY bgcolor="magenta">
<A href="tv:">
<OBJECT data="tv:" width="60%" height="60%" align="center">
</OBJECT>
</A>
<BR>
<P>Welcome to the Day & Night & Day Interactive Experience</P><BR>
<P>Watch below for more information about the current scene!</P><BR>

<SCRIPT LANGUAGE="JavaScript">
function scenechange(imagename)
{
document.sceneimage.src = imagename + ".png";
}
</SCRIPT>

<IMG name="sceneimage" align="center" src="">

</BODY>
</HTML>
--example98203804805
Content-Location: murder.png
Content-Length: 1495
Content-Type: image/png
```

binary resource data for murder.png image
--example98203804805

This data multipart entity, (of total length, including headers of 2400 bytes) would be transmitted via UHTTP, in three packets. The first two packets would contain the original data (each containing 1200 bytes of original data as payload) and the third containing the exclusive-or of the first and second payloads as forward error correction data.

The UHTTP headers for each of the three packets would be as follows:

Field (size)	**Packet 1 value**	**Packet 2 value**	**Packet 3 value**	**Description**
Version (5 bits)	00000	00000	00000	UHTTP version
ExtensionHeader (1 bit)	0	0	0	No extension headers in this example
HTTPHeadersPrecede (1 bit)	1	1	1	HTTP headers precede data
CRCFollows (1 bit)	0	0	0	No CRC Follows
PacketsInXORBlock (1 byte)	3	3	3	Number of packets in each XOR FEC block
Retransmit Expiration (2 bytes)	1800	1800	1800	This will be retransmitted for the next 1800 seconds (this value will decrease as the show progresses)
TransferID (16 bytes)	0x14323ab4123ab4567 cd89ef0567cd89ef0	same	same	UUID for this transmission
Resource Size (4 bytes)	2400	2400	2400	Size of payload
SegStartByte (4 bytes)	0	1200	2400	Offset into the stream where this packet's payload starts

In this example, the 28 UHTTP header bytes for the first packet would be:

0x02, (version, options)
0x03, (packets in XOR block)
0x07, 0x08, (retransmit expiration)
0x14, 0x32, 0x3a, 0xb4, 0x12, 0x3a, 0xb4, 0x56, 0x7c, 0xd8, 0x9e, 0xf0,
0x56, 0x7c, 0xd8, 0x9e, 0xf0, (Resource ID)
0x00, 0x00, 0x09, 0x2e, (Resource Size)
0x00, 0x00, 0x00 (Segment Start Byte Offset)

Following the header in the first packet, would be the first 1200 bytes of the MIME-encoded payload. Following the header in the second packet would be the last 1175 bytes of the MIME-encoded payload. Following the UHTTP header in the third packet would be 1200 bytes, where each byte was the exclusive-or of the corresponding byte offsets in the first two packets.

These packets would then be transmitted repeatedly during the session. The header values for each packet would remain the same, with the exception of the Retransmit Expiration field. The value of this field would decrease as the end of the transmission of the UHTTP packets drew near.

Triggers:

The following trigger would be sent after the data was first transmitted to trigger the beginning of the enhanced television experience:

```
<lid://nicebroadcaster.com/show27/launch.html>[name:Day  &  Night
Day Again Interactive]
```

This trigger content would be encapsulated in a UDP packet and sent to multicast address: *224.0.1.112*, port *52127+1* (as specified by the announcement)

This trigger packet would also be transmitted periodically later on, to allow viewers who tune in late to join in the fun.

Later on, during the program, the content creator might send the following trigger to the same multicast address and port to make the content change to reflect the fact that a murder scene has just begun in the program:

```
<lid://nicebroadcaster.com/show27/launch.html>
("murder")]
```

This trigger would cause the active enhancement page (if it matched the URL in the trigger) to execute the ECMAScript function 'scenechange("murder")', which would cause the murder.png image to be displayed within the page. If the specified URL was not currently being displayed, the trigger would be ignored because this trigger does not include a [name:] attribute,

Near the end of their program, they might send the following trigger to tell their interactive application to shut down. This would allow them to more accurately synchronize with the end of the program, rather than relying on the session timing information in the announcement.

```
<lid://nicebroadcaster.com/show27/launch.html>
[script:window.location="tv:"]
```

Appendix F: References

Document markup language HTML 4.0: http://www.w3.org/TR/REC-html40/

Document scripting language ECMAScript: http://www.ecma.ch/stand/ecma-262.htm

Document Object Model DOM Level 0: http://www.w3.org/DOM

UUIDs and GUIDs (IETF work in progress draft-leach-uuids-guids-01): *The draft is no longer available.*

MPEG-2 CRC (ISO/IEC 13818-1, Annex A: CRC Decoder Model)

TV URLs: *This draft is not longer available.*

Hypertext Transfer Protocol (HTTP) 1.1 (RFC 2068): ftp://ftp.isi.edu/in-notes/rfc2068.txt

Data Delivery via Analog Video VBI: (Working Group): http://www.ietf.org/html.charters/ipvbi-charter.html

Aggregation & encoding of multiple resources into a single resource for delivery:

MIME multipart/related: http://info.internet.isi.edu/in-notes/rfc/files/rfc2387.txt

MIME HTML (rfc2110): ftp://ftp.isi.edu/in-notes/rfc2110.txt

Content description SDP: ftp://ftp.isi.edu/in-notes/rfc2327.txt

Session Announcement Protocol (SAP): http://www.ietf.org/html.charters/mmusic-charter.html

Content encoding: ftp://ftp.isi.edu/in-notes/rfc1951.txt (deflate), and ftp://ftp.isi.edu/in-notes/rfc1952.txt (gzip)

Datagram format IP: ftp://ftp.isi.edu/in-notes/rfc791.txt

Multicast datagram format multicast IP: ftp://ftp.isi.edu/in-notes/rfc1112.txt

Cascading Style Sheets: http://www.w3.org/pub/WWW/TR/REC-CSS1

text/css: ftp://ftp.isi.edu/in-notes/rfc2318.txt

audio/basic: http://www.oac.uci.edu/indiv/ehood/MIME/2046/rfc2046.html

message-id style unique id: ftp://ftp.isi.edu/in-notes/rfc822.txt

Triggers:

EIA-746A Proposal for Sending URLs over EIA 608 T2, available for purchase at the Global Engineering Documents Website: http://global.ihs.com/

UDP (User Datagram Protocol): ftp://ftp.isi.edu/in-notes/rfc768.txt

---

Appendix G: Glossary

Announcements: Announcements are used to announce currently available programming to the receiver.

Binding: In the context of this document, an ATVEF binding is the definition of how the ATVEF transport specifications are encoded on a specific video network standard. (For an example, see the ATVEF Binding to NTSC.)

Content creator: In the context of this document, an ATVEF content creator has the role of originating the content components of the television enhancement including graphics, layout, interaction, and triggers.

CSS1 (Cascading Style Sheets, Level 1): CSS1 is a simple style sheet mechanism that allows content creators and readers to attach style (e.g. fonts, colors and spacing) to HTML documents. The CSS1 language is human readable and writable, and expresses style in common desktop publishing terminology.

Datagram: a block of data that is "smart" enough (actually, which carries enough information) to travel from one Internet site to another without having to rely on earlier exchanges between the source and destination computer.

DHTML (Dynamic HTML): a term used by some vendors to describe the combination of HTML, style sheets, and scripts that enable the animation of web pages.

DOM (Document Object Model): the Document Object Model is a platform- and language-neutral interface that will allow programs and scripts to dynamically access and update the content, structure and style of documents. The document can be further processed and the results of that processing can be incorporated back into the presented page.

ECMAScript: A general purpose, cross-platform programming language.

FEC (Forward Error Correction)

FTP (File Transfer Protocol): A standard for finding and transferring files on the Internet.

HTML (Hypertext Markup Language): a collection of tags typically used in the development of Web pages.

HTTP (Hypertext Transfer Protocol): a set of instructions for communication between a server and a World Wide Web client.

IANA (Internet Assigned Numbers Authority): the central registry for various Internet protocol parameters, such as port, protocol and enterprise numbers, and options, codes and types. The currently assigned values are listed in the Assigned Numbers document. If you'd like more information or want to request a number assignment, you can e-mail IANA at iana@isi.edu.

IETF (Internet Engineering Task Force): the IETF is a large, open community of network designers, operators, vendors, and researchers whose purpose is to coordinate the operation, management and evolution of the Internet, and to resolve short-range and midrange protocol and architectural issues. It is a major source of proposals for protocol standards which are submitted to the IAB for final approval. The IETF meets three times a year and extensive minutes are included in the IETF Proceedings.

IP (Internet Protocol): This protocol is one of the languages computers connected to the Internet use to communicate.

IP multicast: A one-to-many transmission, in contrast to Unicast, Broadcast. An extension to the standard IP network-level protocol. RFC 1112, Host Extensions for IP multicasting, authored by Steve Deering in 1989, laid the groundwork for IP multicasting. The RFC describes IP multicasting as: "the transmission of an IP datagram to a 'host group', a set of zero or more hosts identified by a single IP destination address. A multicast datagram is delivered to all members of its destination host group with the same 'best-efforts' reliability as regular unicast IP datagrams. The membership of a host group is dynamic; that is, hosts may join and leave groups at any time. There is no restriction on the location or number of members in a host group. A host may be a member of more than one group at a time."

ISO (International Organization for Standardization): a voluntary, non treaty organization founded in 1946 which is responsible for creating international standards in many areas, including computers and communications. Its members are the national standards organizations of the 89 member countries, including ANSI for the U.S.

MIME (multipart/signed, multipart/encrypted content-types) a protocol for allowing e-mail messages to contain various types of media (text, audio, video, images, etc.).

NABTS (North American Basic Teletext Specification).

Receiver: In the context of this document, an ATVEF receiver is a hardware and software implementation (television, set-top box, or personal computer) that decodes and presents ATVEF content.

SAP (Session Announcement Protocol): the protocol used for session announcements.

SDP (Session Description Protocol): SDP is intended for describing multimedia sessions for the purposes of session announcement, session invitation, and other forms of multimedia session initiation.

Transport operator: In the context of this document, the transport operator runs a video delivery infrastructure (terrestrial, cable, satellite, or other) that includes a transport for ATVEF data.

Triggers: used to identify the URL and some human-readable string to use in the announcement to the user. In order to announce the availability of the interactive television experience to the user, (as opposed to announcing it to the client downloader mechanism).

TV Enhancement: A collection of Web content displayed in conjunction with a TV broadcast as an enhanced or interactive program.

UDP (User Datagram Protocol): an Internet Standard transport layer protocol defined in

STD 6, RFC 768. It is a connection-less protocol which adds a level of reliability and multiplexing to IP.

UHTTP (Unidirectional Hypertext Transfer Protocol): UHTTP is a simple, robust, one-way resource transfer protocol that is designed to efficiently deliver resource data in a one-way broadcast-only environment. This resource transfer protocol is appropriate for IP multicast over television vertical blanking interval (IPVBI), in IP multicast carried in MPEG-2, or in other unidirectional transport systems.

UUID (Universally Unique Identifier) Also known as GUID (Globally Unique IDentifier) is an identifier that is unique across both space and time, with respect to the space of all UUIDs.

W3C (World Wide Web Consortium): The W3C, an an international industry consortium, was founded in October 1994 to lead the World Wide Web to its full potential by developing common protocols that promote its evolution and ensure its interoperability.

Internet Appliance Design

ENHANCING TV WITH ATVEF

Jason Steinhorn and Mark Kohler

New standards are making the delivery of Web-based and enhanced content alongside television a reality. This article describes the ATVEF enhanced television standard and the requirements for designing ATVEF-compatible receivers.

Despite its failings, the 1996 release of WebTV was the start of a revolution in Web surfing. For the first time, connecting to the Internet was as easy as using a television set and an infrared keyboard. WebTV's first Internet set-top product allowed users to surf the Web on their television screens, getting e-mail and reading news from their sofas and armchairs. Two years later, their WebTV Plus product improved on this paradigm, allowing users to simultaneously view Web content and television, with the broadcast signal embedded in the Web content, in a picture-in-picture window.

The convergence of computers and television has been predicted by technology analysts for many years. Instead of having both PCs and televisions, they suggest many consumers will eventually own only a single device that will have the widespread availability and ease-of-use of television, combined with the interactive power and flexibility of a PC. Along this path, we are already seeing both devices adopting features formerly reserved for the other.

From the PC side, computers are becoming more adept at handling video content. MPEG compression standards have allowed computers to display video; multicast IP and push technologies allow the TCP/IP protocols to simulate a broadcast infrastructure.

From the TV side, we are just starting to see the emergence of standards that allow Web-based content to be broadcast to your television. One of the most popular of these standards is the Advanced Television Enhancement Forum (ATVEF), a specification developed and supported by some of the biggest names in the broadcasting, computer, and consumer electronics industries.

The goal of this article is to discuss the future of television as an

Internet appliance. Because ATVEF is one of the most promising standards in the enhanced television world, it's a good example of where Internet-enhanced TV is going. The bulk of this article will be geared toward describing the ATVEF standard and its technical implementation. For the sake of completeness, we will also discuss how ATVEF might be used in the coming years, its industry support, and its major competitors.

Content specification

In a nutshell, ATVEF is a standard for creating enhanced, interactive television content and delivering that content to a range of television, set-top, and PC-based receivers. ATVEF defines the standards used to create enhanced content that can be delivered over a variety of media, including analog (NTSC) and digital (ATSC) television broadcasts, and a variety of networks, including terrestrial broadcast, cable, and satellite.

By defining the standards used to create enhanced content, the ATVEF specification also defines the minimum functionality required by ATVEF receivers to parse and display this content. One of the major goals of ATVEF was to create a specification that relies on existing and prevalent standards, so as to minimize the creation of new specifications. Not surprisingly, the group chose to base their content specification on existing Internet technologies such as HTML and JavaScript.

Besides minimizing the number of standards that the ATVEF working group needed to create, forcing content creators to base their content on existing Internet technologies provides two other important benefits. First, because the content specifications are fully Web-compatible, millions of pages of potential content already exist. And second, considering how easy it is to use many of today's Web-authoring tools, practically anyone can become an ATVEF content developer.

The ATVEF 1.0 Content Specification mandates that receivers support HTML 4.0, JavaScript 1.1, and Cascading Style Sheets. This is a minimum content specification because all receivers must support these standards, but they are allowed to support others as well—Java and VRML, for example. Establishing a minimum content specification is important to content developers who want to produce the richest content possible, while ensuring that their content is available to the maximum number of viewers.

With ATVEF's membership being much greater on the side of content developers than on set-top box and TV manufacturers, it's no surprise that the minimum standard provides for nearly the same feature set as the latest PC-based web browsers. As more manufacturers consider adopting ATVEF, we are likely to see additional content specifications—perhaps an "ATVEF Lite"—that provide less functionality at a reduced hardware and software cost. This is sure to please companies that design embedded systems, as the majority of embedded web browsers don't yet have the same level of content support as typical PC-based browsers.

Of course, including a Web browser on a television set introduces some

possibilities for exciting new content. To support these, the ATVEF specification calls for new extensions to the existing standards. The most prominent extension to HTML defined by the ATVEF specification is the addition of a "tv:" attribute. The "tv:" attribute specifies the insertion of the television broadcast signal into the content, and may be used in an HTML document anywhere that an image may be placed. Creating an enhanced content page that displays a television channel in some area of the page is as easy as inserting an image into an HTML document.

In addition to defining what ATVEF content looks like, the specification also defines how the content gets from the broadcaster to the receiver, and how the receiver is informed that it has enhancements available for the user to access. The latter task is accomplished with triggers.

Triggers

Triggers are mechanisms used to alert receivers to incoming content enhancements. They are sent over the broadcast medium and contain information about enhancements that are available to the user. Among other information, every trigger contains a standard Universal Resource Locator (URL) that defines the location of the enhanced content. ATVEF content may be located locally—perhaps delivered over the broadcast network and cached to a disk—or it may reside on the Internet, another public network, or a private network.

Besides containing information about where the enhanced content is located, triggers may also contain a human-readable description of the content. For example, a trigger may contain a description like, "Press Browse for more information about this show...," that can be directly displayed by the receiver in order to provide information about the nature of the content to the user. Triggers may also contain expiration information to provide the receiver with contextual information about how long the content should be offered to the viewer and a checksum to ensure the integrity of the delivered information.

Lastly, triggers may contain JavaScript fragments. These script fragments (often just single method calls) can trigger execution of JavaScript within the associated HTML page, and can be used for such things as synchronization of the enhanced content with the video signal and updating of dynamic screen data.

Transports

Besides defining how ATVEF content is displayed and how the receiver is notified of new content, the specification also defines how content is delivered. Because your television or set-top box may or may not have a connection to the Internet, the ATVEF specification describes two distinct models for delivering content. These two content delivery models are commonly referred to as transports, and the two transports defined by ATVEF are referred to as Transport Type A and Transport Type B.

Transport Type A is defined for ATVEF receivers that maintain a

connection (commonly called a back-channel or return path) to the Internet. Generally, this network connection is provided by a dial-up modem but may be provided by any type of bi-directional access channel. Transport Type A is a method for delivering only triggers, without additional content. Because there is no content delivered with Transport Type A, all data must be obtained over the back-channel, using the URL(s) passed with the trigger as a pointer to the content.

Transport Type B provides for delivery of both ATVEF triggers and the associated content via a broadcast network. In this model, the broadcaster pushes content to a receiver, which will store it in case the user chooses to view it. Transport Type B uses announcements sent over the network to associate triggers with content streams. An announcement describes a content stream and may include information regarding bandwidth, storage requirements, and language (enhancements may be delivered in multiple languages).

Since a Type B receiving device will, in most cases, need to store any content that will be displayed, it uses announcement information to make content storage decisions. For instance, if a stream requires more storage space than a particular receiver has free, the receiver may elect to discard some older content, or it may elect not to store the announced stream. A drawback of this model is that if a person chooses to start watching a show near the end, there may not be time for the content to be streamed to the receiver, and the person will not be able to view some or all of the content.

To review, the two types of ATVEF data are triggers and content. If the receiving device has a backchannel to the Internet, Transport Type A will broadcast the trigger only (akin to a URL), and content will be pulled over the Internet. If the receiving device doesn't have an Internet connection, Transport Type B allows both the triggers and content to be delivered over the broadcast channel.

Delivery protocols

The ATVEF specification also defines a reference protocol stack used for content delivery. While all of the high-level protocol layers are well-defined for every ATVEF implementation, the link layer and physical layer protocol layers are dependent on the broadcast network. This is obvious when you consider that it is not possible to transmit analog data over cable the same way you would transmit digital data over satellite. Figure 1 illustrates a standard ATVEF protocol stack for delivery of enhanced content.

For traditional bi-directional Internet communication, the Hypertext Transfer Protocol (HTTP) defines how data is transferred at the application level. But because one can't have a two-way connection over a broadcast medium, we require a unidirectional application-level protocol for data delivery. ATVEF defines this protocol to be the Unidirectional Hypertext Transfer Protocol (UHTTP). UHTTP is based on UDP, as opposed to TCP. This makes sense, of course, because UDP is a connectionless protocol suitable for a broadcast network.

Like HTTP, UHTTP uses traditional URL naming schemes to reference content. Therefore, content creators can reference enhancement pages using the standard "http:" and "ftp:" naming schemes. To this, ATVEF adds the "lid:" or local identifier URL naming scheme. The "lid:" naming scheme allows content creators to reference content that exists locally (on the receiver's memory or disk drive, for example) rather than the Web.

With HTTP, as well as with many other Internet application protocols, the TCP layer provides error detection and re-transmission facilities. But for a unidirectional protocol, there is no possibility for retransmission requests. Thus, UHTTP must implement error correction without retransmission, sometimes called Forward Error Correction (FEC). Using sophisticated FEC algorithms, if the data is not too badly corrupted, it can be regenerated with only the received information. With their emphasis on error correction instead of detection, the coding schemes used in unidirectional communications are more similar to the algorithms used in data storage like digital tapes and CD-ROMs, than those used in traditional bi-directional communications.

Bindings

How ATVEF data is delivered over a particular network—from the network layer protocol down to the physical layer—is called the binding. In order for ATVEF to provide interoperability between broadcast networks and receivers, it's important that each physical network have only one binding. And it is equally important that each binding provide a fully comprehensive definition of the interface between the broadcast network specification and the ATVEF specification.

At this point, ATVEF has defined bindings for delivering data over IP multicast as well over NTSC. Because the transmission of IP is defined (or can be) for virtually every type of television broadcast network, the binding to IP is considered the reference binding. So, defining an ATVEF binding for a new network could be as easy as describing how to run IP over that network. Figure 1 illustrates the protocol stack for the reference binding.

ATVEF over NTSC

NTSC is the standard for analog television broadcasts in the U.S. Unless you have an HDTV set already, the televisions in your home are nothing but NTSC receivers. Part of the NTSC standard defines a frame (image) as consisting of 525 horizontal lines, each line drawn (or scanned) left to right. During a screen scan, only every other line is drawn; therefore, it takes two full screen scans to draw a single frame.

Each time the electron gun in the television's cathode ray tube finishes scanning a half-frame, it must return to the upper left-hand corner of the television screen to prepare for the next half-frame. This takes a non-trivial amount of time, so each movement of the electron gun must be re-synchronized with the incoming signal. This is done by adding a set of unused lines of data to the end of each screen scan, giving the electron gun time to return to its starting position. These 21 extra

"lines" make up what is called the vertical blanking interval (VBI). (If you want to see the VBI for yourself, fiddle with the vertical hold knob on your TV and look for a horizontal black stripe across your screen.)

As it turns out, only the first nine lines of the VBI are actually required to reposition the cathode ray. This leaves 12 more lines that can be used to broadcast data. In fact, in the U.S., closed captioning data has been broadcast on line 21 for many years. Each line of the VBI has a transmit rate of about 17K/sec. So in theory, the VBI associated with each NTSC-encoded television channel could carry up to 204K/sec (12 lines at 17K/sec per line) of piggy-backed data. However, after taking into account the overhead associated with the various protocol layers and the need to prevent conflicts with closed captioning and other data already broadcast within the VBI space, the maximum achievable rate for ATVEF data transmission is somewhat lower than this—probably around 100K/sec.

Transport Type A. The Type A transport binding for NTSC is easy to describe. ATVEF triggers are simply broadcast in line 21 of the VBI. For purposes of data integrity, the NTSC binding for Transport Type A requires that each trigger contain a checksum. The binding also recommends that the trigger length not exceed 25% of the total bandwidth of the line, in order to avoid conflicts between triggers, closed captioning data, and data from any future services that might also use line 21.

While ATVEF triggers could have been placed on some other line of the VBI, placing them on line 21 has advantages for receiver manufacturers. For example, most standard NTSC video decoder chips already have the ability to extract line 21 of the VBI (for closed captioning support). By placing triggers in that same line, hardware manufacturers are not forced to upgrade to more expensive decoders that support data extraction in other lines of the VBI.

Transport Type B. In addition to sending triggers on line 21 of the VBI, the Transport Type B NTSC binding includes a mechanism for delivering IP datagrams over the other VBI lines. IP over VBI (IP/VBI) is an Internet Draft of the Internet Engineering Task Force (IETF). As such, IP/VBI is not yet a standard, just a work in progress. Therefore, some details of some of the encapsulation, compression, and error detection schemes may change, but the architecture is unlikely to change radically. Figure 2 illustrates the protocol stack defined by ATVEF down to the IP layer, and defined by IP/VBI below that.

At the bottom of the stack is the NTSC television standard. At the lowest level, the television signal transports NABTS (North American Basic Teletext Standard) packets. NABTS is a method of modulating data onto the VBI. A typical NABTS packet gets encoded onto a single VBI line. NABTS, by way of its own forward error correction, supports correction of all single-bit, double-bit, and single-byte errors, as well as the ability to regenerate an entire missing packet. The NABTS packets are removed from the VBI to form a sequential data stream. This data stream—encapsulated in a SLIP-like protocol—is unframed to produce IP packets, which are handled equivalently across all ATVEF network

types that implement the IP reference binding.

As you can see, a specific network binding is not complicated, but is detailed enough (the full IP/VBI draft standard is obviously much more detailed than what we've presented) that anyone creating a broadcast network or building an ATVEF receiver has enough information to make their design ATVEF-compliant. And, while we've only presented the NTSC binding here, there are—or soon will be—well defined ATVEF bindings to every other major video network standard, including PAL and SECAM (the European counterparts to NTSC), ATSC (digital terrestrial broadcast), cable, and satellite.

Design issues

Even for those intimately familiar with the specification, implementing an ATVEF receiver is not a trivial chore. Because the specification is flexible with respect to many of the implementation details, the embedded software developers—and, in some cases, the hardware designers—have to determine exactly how the receiver will be integrated with the television or the rest of the set-top box.

The first major decision when designing an ATVEF receiver is whether to support Transport Type A or B. Often, this decision is driven by the type of network the receiver will be connected to. For a satellite television set-top box that provides no backchannel to the Internet, the obvious decision is to support Type B. But for a cable television set-top box that doubles as a cable modem with dedicated Internet access, it may be okay to support only Type A. Of course, choosing to support a high-bandwidth option like Transport B will also require additional hardware and/or software performance.

As a typical example, let's suppose that we were building a set-top box that would serve as an NTSC receiver with ATVEF support. Assuming the standard NTSC binding for Type B—NABTS encoding of the data in the VBI—we must decide how we will decode this data when received. The most obvious choice is to use an NTSC video decoder that will parse all VBI lines in hardware. But, as we mentioned earlier, while some of the higher-end decoders support this functionality, these decoders tend to be a little more expensive; when building millions of set-top boxes, every penny saved can make a big difference in the bottom line.

The other option is to do the NABTS decoding in software. Unfortunately, software decoding is processor intensive. In fact, some benchmarks have indicated typical VBI decoding requires up to 2% of a Pentium-class 166MHz processor per VBI line. For full decoding of VBI lines 10 through 20, this would require about 20% of that same processor's time. Of course, these specific issues are only related to NTSC receivers. ATVEF receivers on digital—or non-NTSC analog—networks have a whole set of different issues that must be addressed.

Another major design issue that developers must consider is user interface. By design, the ATVEF specification puts no restriction on how triggers and data are presented to the user. Implementers must decide

how these things are done. For example, it seems reasonable that the user should be able to decide if he would like to receive indication of incoming content or not. But nothing in the specification dictates that implementers must allow users to turn off enhancements.

So what?

So now you're wondering, "What is this enhanced TV stuff going to do for me?" The most obvious answer, unfortunately, is that it is going to try to entice you to spend money. For decades, the television has been used to solicit your hard-earned cash through a seemingly endless stream of commercials. But, despite the annoying jingles that we can't get out of our heads, the slogans that pervade pop-culture, and the famous spokespersons who just won't go away, television advertising has always lacked the ability to "complete the transaction." Never before has television advertising had the means to allow the viewer to make a spontaneous purchase, to buy with the click of a button. Now it does.

That doesn't just mean that every commercial will include a "BUY ME NOW" button. It also means that you'll be able to make purchases during your favorite shows. Clicking on Dan Marino's football jersey during Monday Night Football may pop up a description of the collectible garment, with an opportunity to purchase one right away. Or, during that same Monday Night Football game, the network may offer you the option to receive alternate camera footage, live from the home team's clubhouse or from the camera on the referee's shirt—for a fee, of course.

But let's not just focus on the advertising; enhanced television has the ability to improve your viewing experience as well. Imagine interactive game shows, where the contestants are chosen during the show to participate directly from their living rooms. Or you're watching MTV, and with the click of a button, are finally able to get the lyrics to that ridiculous song you can't stop humming. Imagine choose-your-own-ending television shows where viewers have the option to vote on which of a variety of outcomes will happen.

And not only will television provide enhanced content, it will also have the means to provide personalized content. Take regular NTSC broadcasts, for example. VBI data (and hence ATVEF data) can be added to an NTSC signal at any point, and even more than one point, in the path the signal travels from the broadcaster to the receiver. Therefore, a broadcaster could insert ATVEF content on a national scale, a local cable operator could add ATVEF content relating to local markets, and an automated profiler in your receiver can figure out which specific content would most appeal to you, and display it. National news broadcasters will now have the ability to provide local headlines, or better yet, headlines that appeal specifically to you.

Industry support and competing standards

Enhanced television is not a new idea. For decades, companies have built visions of enhanced television and tried to sell their visions to

advertisers and consumer electronics manufacturers. However, none of these proprietary systems has caught on. Enhanced television has a "chicken and egg problem"—broadcasters are reluctant to invest in enhanced television content and infrastructures before the consumer electronics companies can guarantee a reasonably sized audience. And the consumer electronics companies find it difficult to sell enhanced TV receivers without the support of the broadcasters, who must provide enhanced content.

Today, two main standards compete in the area of enhanced television: ATVEF and Broadcast HTML. Broadcast HTML was created from ATSC-related work to develop the DTV Application Software Environ-ment (DASE). It's a combination of an XML-based subset of HTML 4.0, along with a Java Virtual Machine and Sun's PersonalJava API.

Both standards have significant industry support, and neither is likely to disappear soon. That leaves broadcasters, hoping to avoid a prolonged "VHS vs. Beta" fight, worried. Many are looking to the ATVEF and DASE members to reconcile their differences, or provide a minimum level of interoperability between the two standards.

Some companies are not waiting for the standards to settle. CNN, Discovery Channel, and HBO are among a handful of broadcasters already delivering enhanced content on a regular or semi-regular basis. In fact, each week over 1,000 hours of network, syndicated, and cable TV programming include content enhancements. Consumer electronics companies are designing their next-generation set-top boxes to comply with enhanced television specifications. And embedded web browser companies are already providing enhanced television support in their browsers. You can be sure that enhanced television is coming.

Jason Steinhorn is an embedded software engineer at Hughes Network Systems in Gaithersburg, MD. He is currently designing and developing a Web-enabled satellite television set-top box. Jason can be reached at jsteinhorn@hns.com.

Mark Kohler develops software for broadband network equipment at Nomadix, in Santa Monica, CA. He can be reached at mkohler@nomadix.com.

Acknowledgments

The authors wish to thank David Mott of Liberate Technologies for reviewing this article for technical accuracy. David has served on the ATVEF Technical Working Group since its inception. He can be reached at mott@liberate.com.

Reference links

The following list of World Wide Web links may prove helpful should you wish to further research the topics presented in this article.

http://www.atvef.com/—Official ATVEF site

www.atvef.com/library/spec.html—Most recent ATVEF technical specifications

toocan.philabs.research.philips.com/misc/atsc/bhtml/—Broadcast HTML standard

www.ietf.org/internet-drafts/draft-ietf-ipvbi-nabts-05.txt—IP over VBI Internet Draft

http://www.liberate.com/—Information about Liberate's TVNavigator embedded television browser

www.microsoft.com/tv/default.asp—Information about Microsoft's TVPAK browser software

http://www.webtv.net/—Information about WebTV Networks

http://www.atsc.org/—Advanced Television Systems Committee (ATSC)

Return to Internet Appliance Design Home Page

Return to Embedded.com

United States Patent & Trademark Office

Office of Initial Patent Examination -- Scanning Division

Application deficiencies found during scanning:

☐ Page(s)____________of________________________________ were not present for scanning.
(Document title)

☐ Page(s)____________of________________________________ were not present for scanning.
(Document title)

Pages 5 - 52 as part of the specification are exhibits.

☐ ***Scanned copy is best available.***